(Kunst)Matige intelligentie

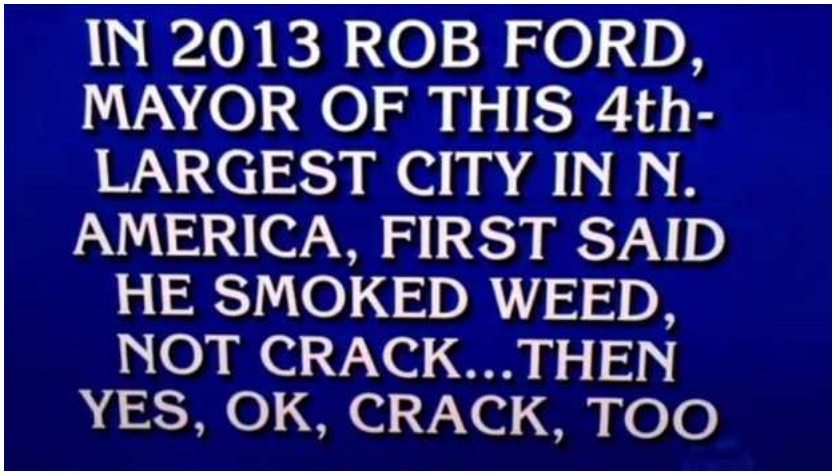
programmeren, α-β, nonogrammen, Tetris

dr. Walter Kosters, Informatica

β-dag Leiden, donderdag 11 februari 2016

www.liacs.leidenuniv.nl/~kosterswa/


$100 $100 $100 $100 $100 $100
$200 $200 $200 $200 $200 $200
$300 $300 $300 $300 $300 $300
$400 $400 $400 $400 $400 $400
$500 $500 $500 $500 $500 $500


IN 2013 ROB FORD, MAYOR OF THIS 4th-LARGEST CITY IN N. AMERICA, FIRST SAID HE SMOKED WEED, NOT CRACK...THEN YES, OK, CRACK, TOO


$2,000
$5,000
$5,000
ken
WATSON
BRAD

2011


What is
Toronto????


$1,000,000
WATSON

Bij de studie Informatica (in Leiden) krijg je per jaar een tiental vakken: de colleges. De propedeuse, het eerste jaar van de driejarige bachelor, ziet er als volgt uit:

najaar	voorjaar
Programmeermethoden	Algoritmiek
Fundamentele informatica 1	Logica
Digitale technieken	Databases
Studievaardigheden	Programmeertechnieken
Wiskunde 1	Wiskunde 2

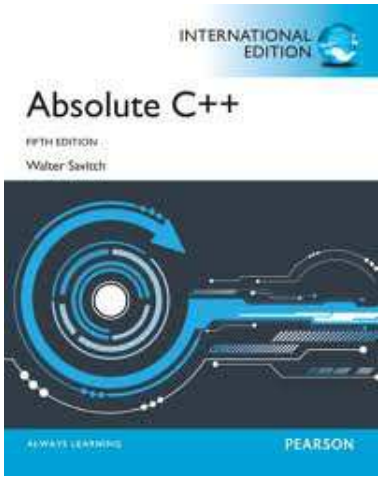
Varianten: Informatica, Informatica & Biologie, Informatica & Economie.

Tweede en derde jaar zijn als volgt:

najaar	voorjaar
Datastructuren	Kunstmatige intelligentie
Fundamentele informatica 2	Complexiteit
Computerarchitectuur	Operating systemen
Programmeertalen	Statistiek
Visualisatie	
Literatuur	Wetenschap & onderzoek
Data mining	Fundamentele informatica 3
Software engineering	
Vak 1	Vak 4
Vak 2	Vak 5
Vak 3	Bachelorproject

Programmeren

Je programmeert een computer in een speciale computer-taal of programmeertaal, zoals C++ of Python.



In Leiden leren alle eerstejaars studenten Informatica, Wiskunde, Natuurkunde en Sterrenkunde programmeren. Voorkennis is niet echt nodig.

Een eerste C++-programma:

```
#include <iostream>
using namespace std;
int main ( ) {
  cout << "Vandaag betadag ..." << endl;
  return 0;
}//main
```

Dit programma zet alleen een tekstje op het beeldscherm.

Let op de — vooral voor mensen nuttige — layout.
En op hoofdletters en kleine letters.

Een tweede C++-programma:

```
// dit is een simpel programma
#include <iostream>
using namespace std;
int main ( ) {
  int getal = 42;  // een variabele
  cout << "Geef een geheel getal .. " << endl;
  cin >> getal;
  cout << "Kwadraat is: "
       << getal * getal << endl;
  return 0;
}//main
```

. . . **I work 9–5 in a 7–11**

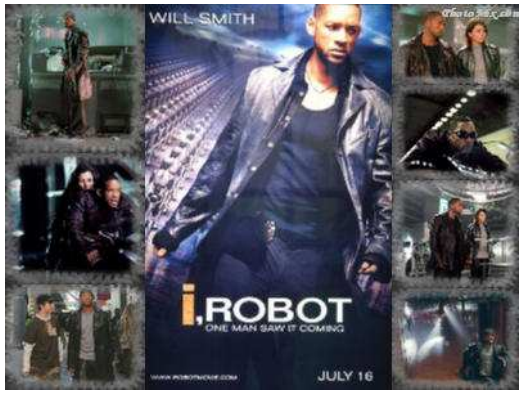
De gebruiker moet zijn/haar geboortejaar als getal invoeren, en daarna geboortemaand en geboortedag. Het programma berekent dan de bijbehorende dag van de week.

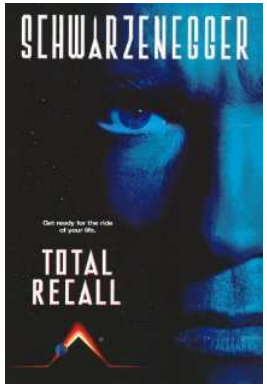
. . . **3/4.1, 0.3, 4.4, 6.6, 8.8, 10.10, 12.12**

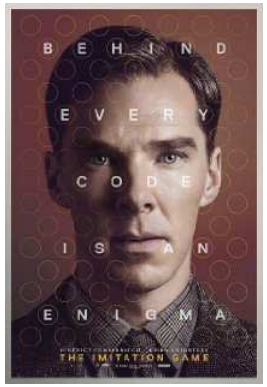
Bijvoorbeeld: 9 november 1989
was een donderdag.

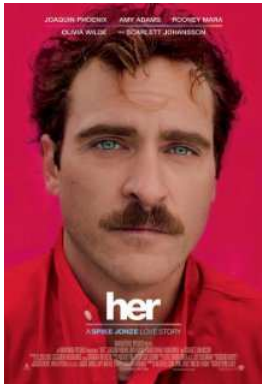
Let op schrikkeljaren;
en 1752?

Kunstmatige intelligentie









Kunstmatige intelligentie (AI, Artificial Intelligence) is een verzamelnaam voor een breed vakgebied, met vragen als:

- *robotica*: Hoe programmeer je een robot?
- *data mining*: Welke films vind je leuk?
- *rechtspraak*: Word je volautomatisch be/veroordeeld?
- *vertalen*: "the spirit is willing but the flesh is weak" $\rightarrow$ ... $\rightarrow$ "the vodka is good but the meat is rotten"?
- *computer games*: Hoe speelt de computer bij CoD?
- *neurale netwerken*: Kun je beurskoersen voorspellen?


IMDb
Showtimes Near You
Oz the Great and Powerful (2013)
Top Billed Cast
Top Billed Crew
Director
Sam Raimi
Monsters University
Trailer #2
In Theaters
Showtimes Near You
Recommended For You
Latest Trailers
Jennifer Lawrence
Oz the Great and Powerful (2013)
2 hrs 10 mins
A small-time magician arrives in an enchanted land and is forced to decide if he will be a good man or a great one.
Metascore
Watching Options
Details

Je kunt op minstens twee manieren naar Kunstmatige intelligentie kijken:

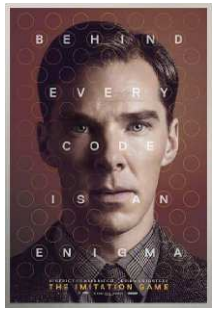
1. vanuit een meer *psychologische* of *filosofische* richting:
   Wat is het verschil tussen een mens en een computer?
   Kan een computer denken (zwemmen, vliegen, . . .)?

2. vanuit een meer *technische* richting:
   Hoe werkt een schaakprogramma?
   Hoe werkt een Marsrobot?

"Do androids dream of electric sheep?" $\longrightarrow$

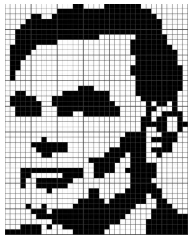
The imitation game

Kunstmatige intelligentie laat computers zich zo gedragen dat het intelligent zou heten als mensen het op die manier zouden doen.



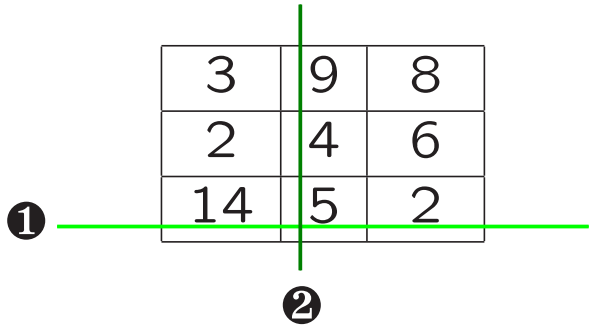
De beroemde Turing-test (1950) is:

> In een afgesloten kamer bevindt zich een mens *of* een computer, waarmee we alleen via toetsenbord en beeldscherm contact hebben.
> Is het een mens of juist een computer?



Het originele probleem was overigens met man $\leftrightarrow$ vrouw.

Maxi en Mini spelen het volgende eenvoudige spel: Maxi wijst eerst een (horizontale) rij aan, en daarna kiest Mini een (verticale) kolom:



Bijvoorbeeld: Maxi ❶ kiest rij 3, daarna kiest Mini ❷ kolom 2; dat levert einduitslag 5.

Maxi wil graag een zo groot mogelijk getal, Mini juist een zo klein mogelijk getal.

Hoe spelen we dit spel zo goed mogelijk?

Als Maxi rij 1 kiest, kiest Mini kolom 1 (levert 3); als Maxi rij 2 kiest, kiest Mini kolom 1 (levert 2); als Maxi rij 3 kiest, kiest Mini kolom 3 (levert 2). Dus kiest Maxi rij 1!

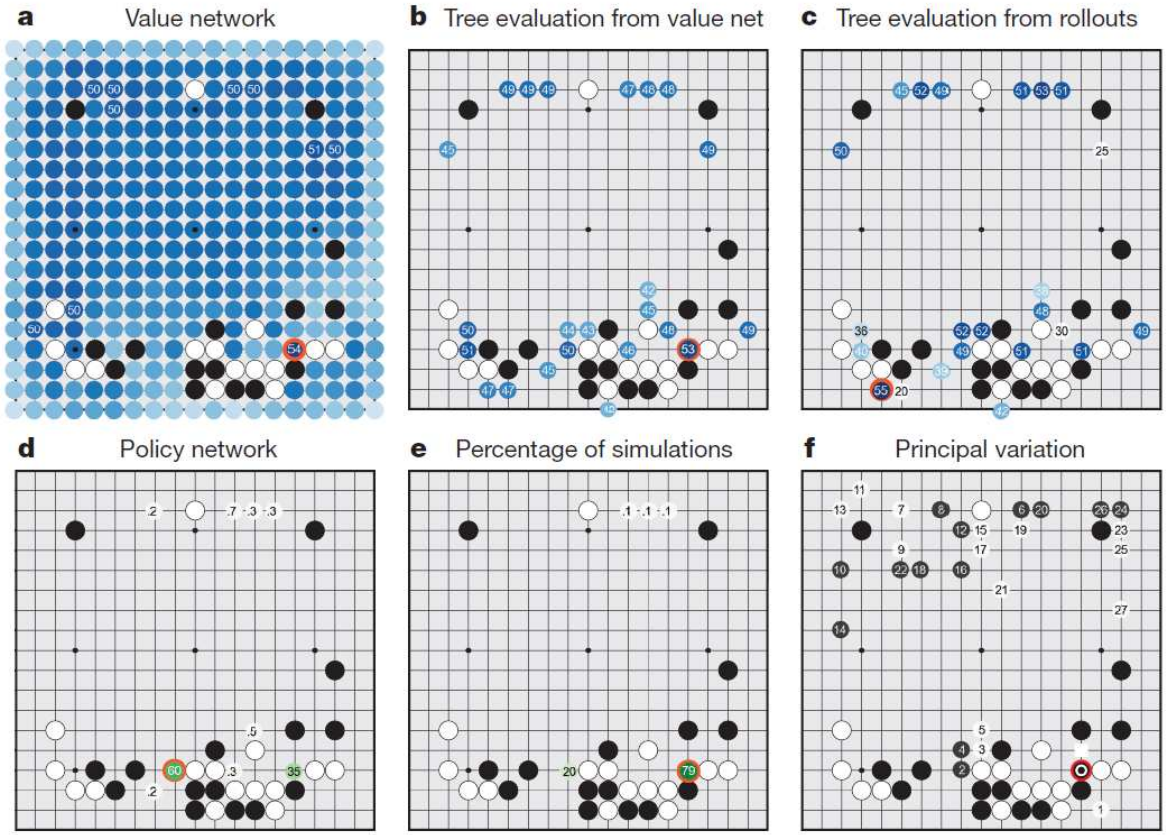
3	9	8
2	**?**	**?**
14	5	2

Nu merken we op dat de analyse hetzelfde verloopt als we niet eens weten wat onder de twee vraagtekens zit.
Het α-β-algoritme onthoudt als het ware de beste en slechtste mogelijkheden, en kijkt niet verder als dat toch nergens meer toe kan leiden.

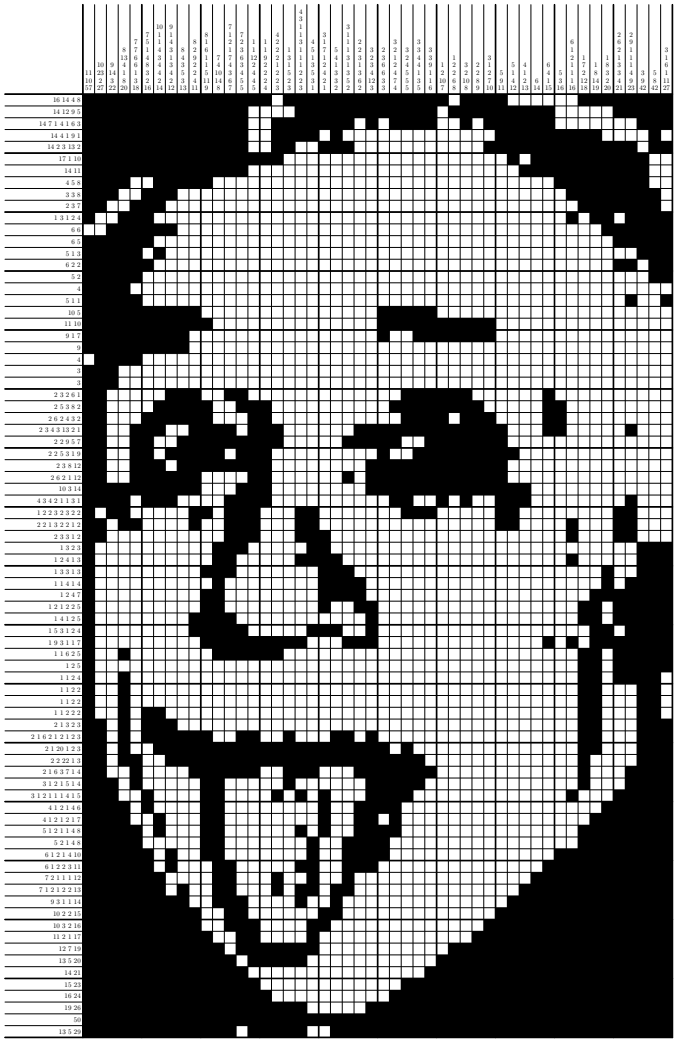
Ieder schaakprogramma gebruikt deze methode.

Schaken

1997



januari 2016: computerprogramma verslaat professional



Als je Japanse puzzels zegt, denkt iedereen aan Sudoku.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

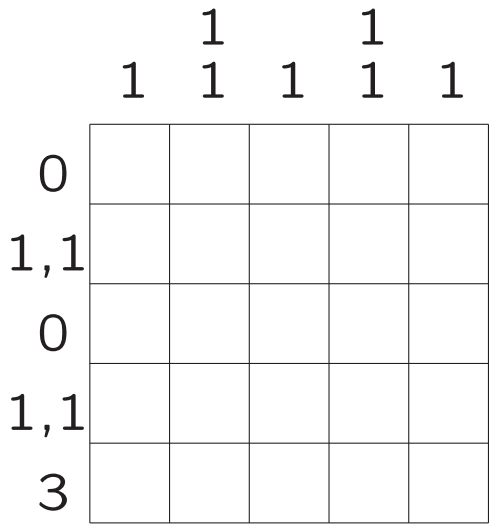
Als je Japanse puzzels zegt, denkt iedereen aan Sudoku.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

bron: Wikipedia

Maar wij gaan het hebben over Nonogrammen.

Een Nonogram is een puzzel; een klein voorbeeld:

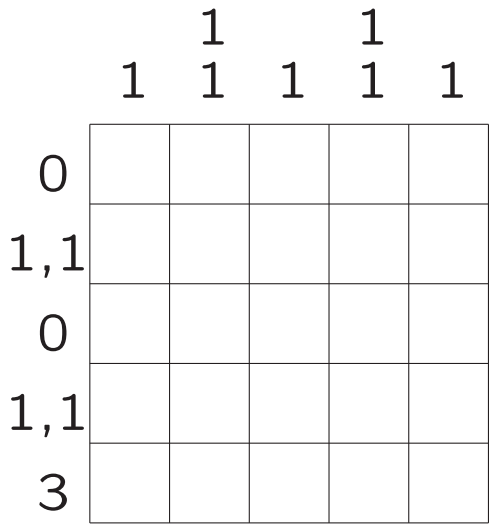


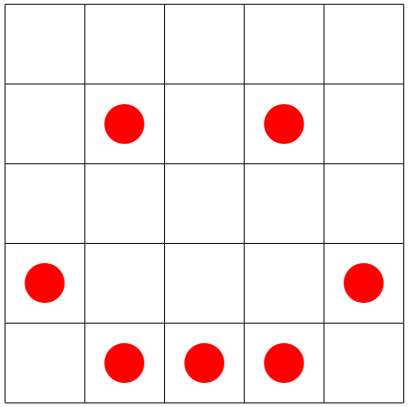
Naast iedere rij en boven iedere kolom staan in volgorde de lengtes van aaneengesloten series rode (of zwarte) vakjes.

Waar moeten die rode vakjes komen?

Voorbeeld — oplossing

De oplossing ziet er zo uit:





Naast iedere rij en boven iedere kolom staan in volgorde de lengtes van aaneengesloten series rode (of zwarte) vakjes.

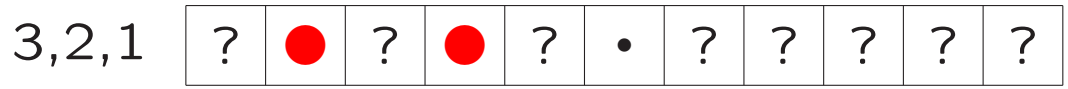
Hoe los je Nonogrammen op?

De meeste mensen gebruiken logische regels, en heuristieken = vuistregels zoals "redeneer eerst een keer via de rijen, en dan via de kolommen".

Een voorbeeld van een logische regel is: "als het getal 3 naast een rij van breedte 5 staat, moet het middelste vakje wel rood zijn". Je kijkt dan eigenlijk naar één rij of kolom.

Eén rij of kolom?

Stel dat je van een rij al weet:

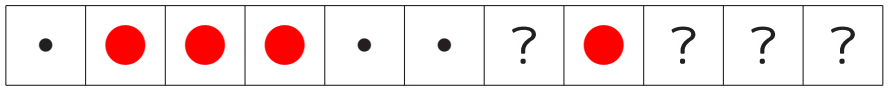


Een • betekent een zeker leeg vakje, een ● staat voor een zeker gevuld vakje. De rest is nog onbekend.

Wat kun je hier nu concluderen?

Eén rij of kolom!

We concluderen dan dat voor deze rij geldt:

3,2,1 

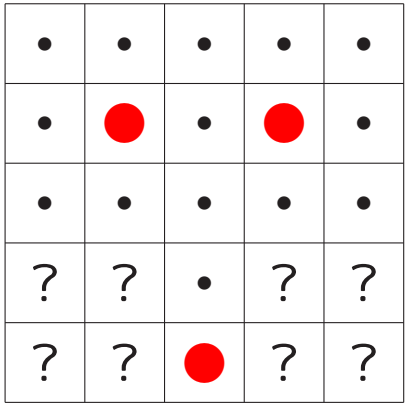
Een • betekent een zeker leeg vakje, een ● staat voor een zeker gevuld vakje. De rest blijft nog onbekend.

Dus door naar een enkele rij of kolom te kijken kun je vooruitgang boeken. En dat gaat goed met dynamisch programmeren.

Voorbeeld — oplossen

Hoe ver komen we als je alleen per rij/kolom kijkt? Een • betekent weer een zeker leeg vakje, een ● staat voor een zeker gevuld vakje.

		1 1		1 1	
	1	1	1	1	1
0					
1,1					
0					
1,1					
3					

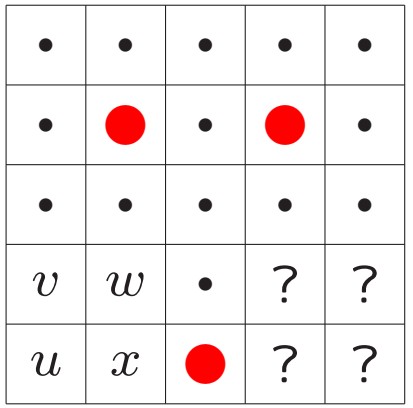


Maar nu zitten we vast . . . tenzij we rijen en kolommen *samen* bekijken.

Voorbeeld — verder oplossen

Dit hadden we:

		1 1		1 1	
	1	1	1	1	1
0					
1,1					
0					
1,1					
3					



Stel dat $u =$ ●, dan (kolom) moet v leeg zijn, en dus (rij) $w =$ ●, en dus (kolom) moet x leeg zijn. Tegenspraak (rij)! Dus u moet leeg zijn. Kortom: potlood & gum!

Dat was een lastige logische redenering, ook voor een computer. Maar de rest is nu eenvoudig.

Een 5×5 Nonogram heeft

$$2^{25} = 2^{10} \cdot 2^{10} \cdot 2^5 = 1024 \cdot 1024 \cdot 32 \approx 32 \text{ miljoen}$$

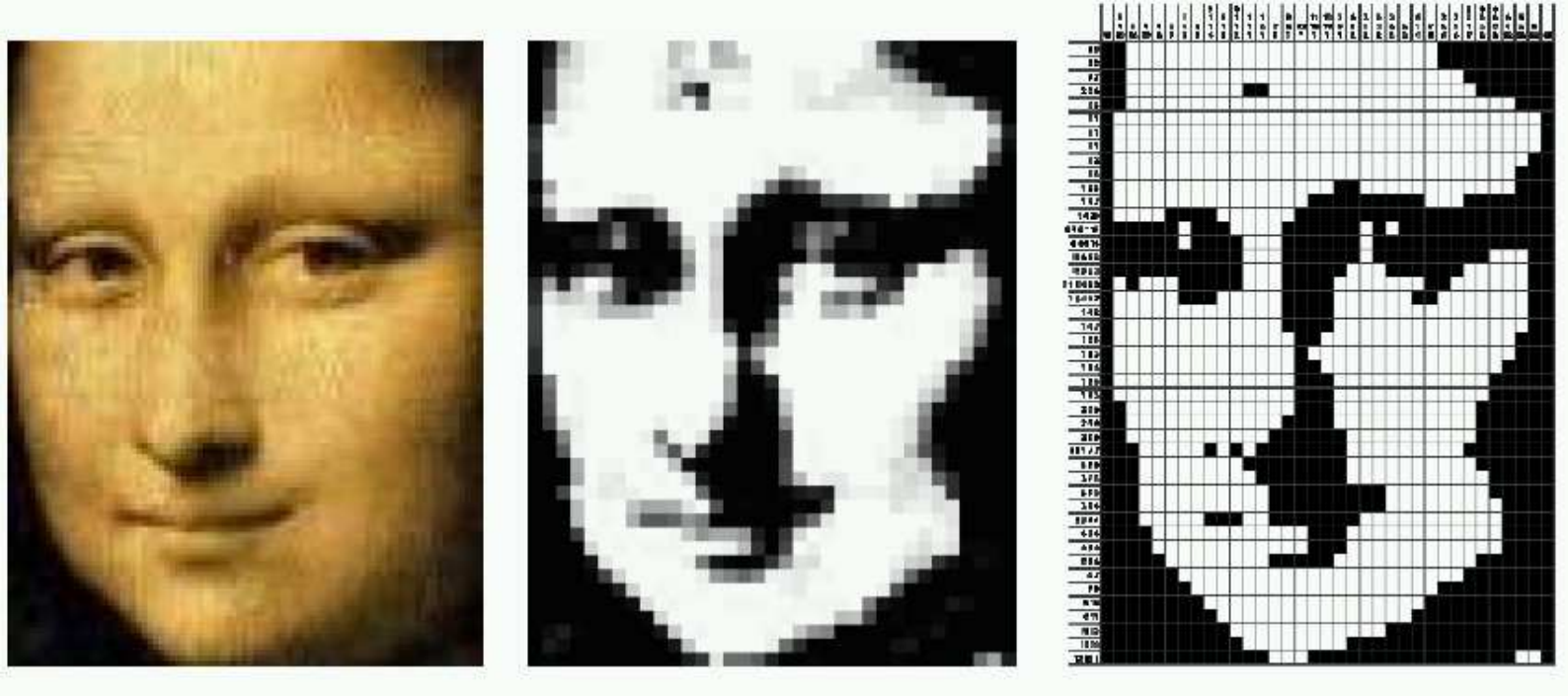
mogelijke invullingen! Want er zijn $5 \times 5 = 25$ vakjes met elk 2 mogelijkheden.

De "80×50 Einstein" heeft $2^{4000} \approx 10^{1200}$ mogelijkheden.

Dus brute-force, alles domweg proberen, lost een complete puzzel niet *snel* op . . .

Dit heeft te maken met het grootste open informatica-probleem $\boxed{\mathcal{P} \stackrel{?}{=} \mathcal{NP}}$. Je kunt \$ 1.000.000 verdienen als je dit oplost!

Hoe maak = construeer = ontwerp je zelf een Nonogram?



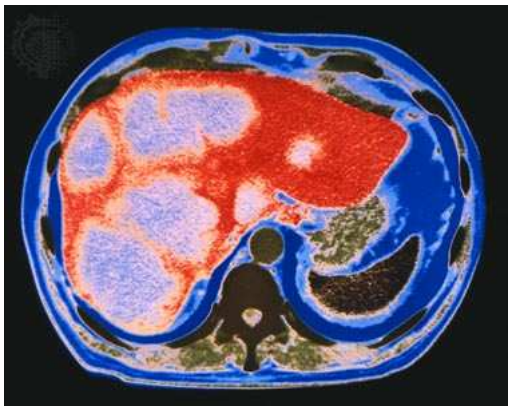
kleurenfoto grijswaarden-plaatje puzzel

www.liacs.leidenuniv.nl/~kosterswa/nono/

Waarom doen wetenschappers Nonogrammen?

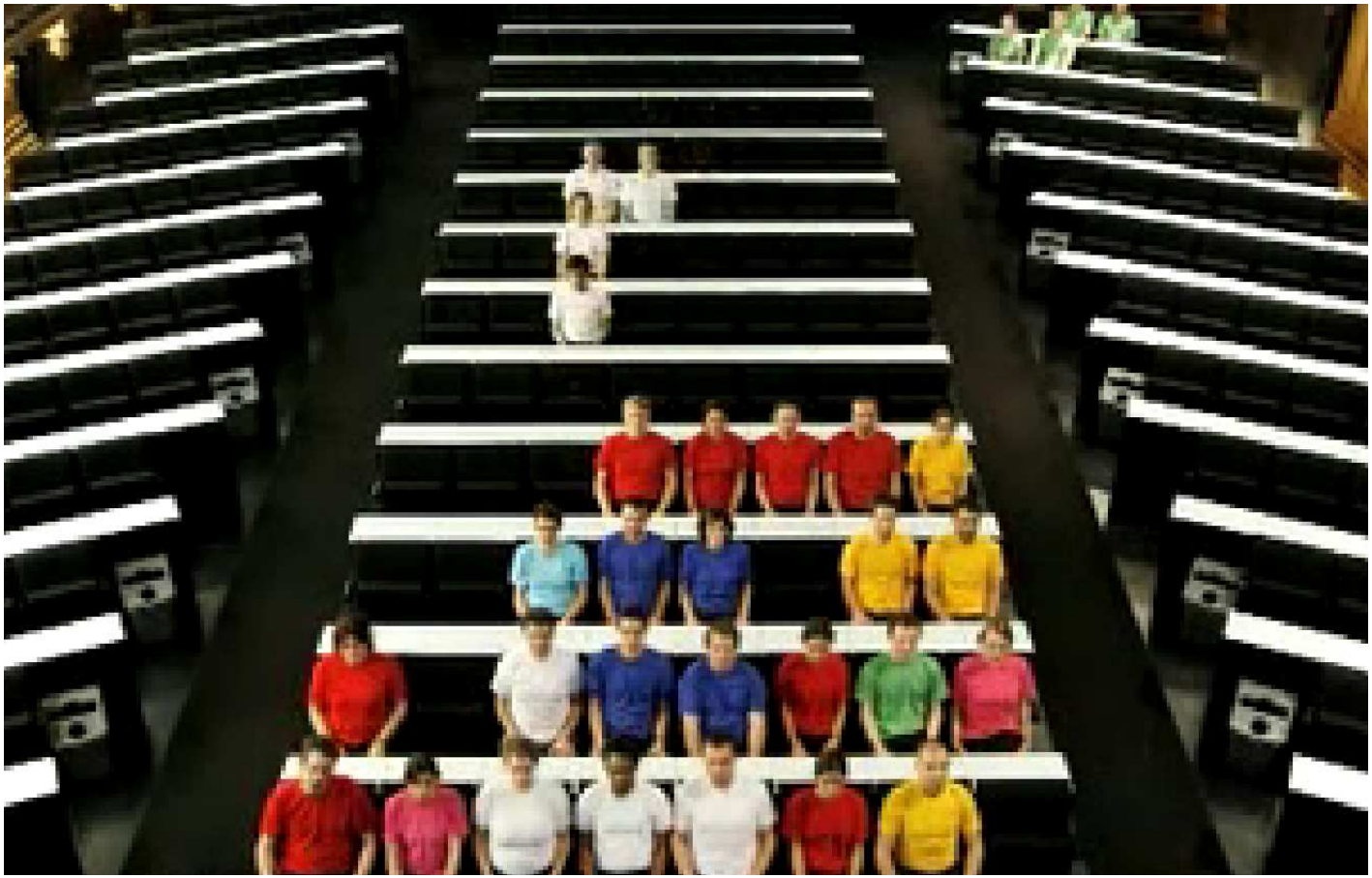
Tomografie houdt zich bezig met het volgende probleem:
Hoe reconstrueer je een object uit projecties?

Voorbeelden:



- Nonogrammen oplossen
- Hoe zien onze organen eruit, gegeven CT-scans?
- Waar zitten de “gaten” in een diamant?

Tetris



YouTube

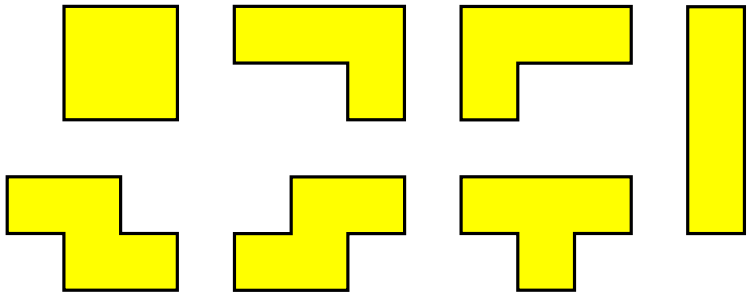
Ook aan een spel als Tetris kleven allerlei vragen:

- Hoe speel je het zo goed mogelijk? (AI)
- Hoe moeilijk is het? (complexiteit)
- Wat kan er allemaal gebeuren?

Zo is bijvoorbeeld bewezen dat sommige Tetris-problemen NP-volledig zijn, dat je bijna alle configuraties kunt bereiken, maar dat niet alle problemen "beslisbaar" zijn, zie:

www.liacs.leidenuniv.nl/~kosterswa/tetris/

De 7 Tetris-stukken:

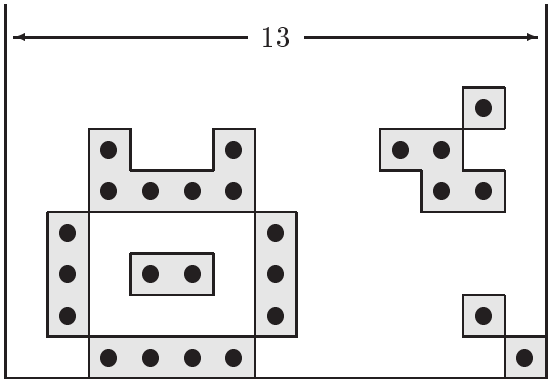


Stukken vallen random; volle regels worden verwijderd.
De vraag "Kun je met een gegeven serie (inclusief volgorde) van deze stukken een bord helemaal leeg spelen?" is NP-volledig.

Als iemand het bord leeg speelt kun je dat eenvoudig controleren. Als het *niet* kan, kan men (tot nu toe) niks beters verzinnen dan alle mogelijkheden één voor één na te gaan!

Alles bereikbaar?

Een “willekeurige” configuratie:



Deze kan gemaakt worden door 276 *geschikte* Tetris-stukken op de juiste plaats te laten vallen.
Let op: alleen geheel gevulde regels verdwijnen, alles daarboven zakt *één rij*.

Claim: op een bord van oneven breedte kan elke configuratie bereikt worden!

Vragen?